Creative Programming for Young Minds

. . . on the TI-99/4A ™

```
300 FOR X=K1 TO 7
310 READ A,B,C
320 CALL COLOR(A,B,C)
330 DATA 9,3,3,10,7,16,11,5,
16,16,5,16,12,7,16,15,7,16,1
4,8,8
340 NEXT X
350 DIM P$(12,32)
360 FOR N=K1 TO 25
370 READ C,C$
380 CALL CHAR(C,C$)
390 NEXT N
400 DATA [illegible]
[illegible]
```

Volume II

CREATIVE
PROGRAMMING INCORPORATED
A SUBSIDIARY OF R.V. WEATHERFORD CO.

CREATIVE Programming for Young Minds

CREATIVE Programming for Young Minds didn't just happen. It represents the harvested fruit of an idea planted several years ago by Dr. Henry A. Taitt. He saw the pressing need for an enrichment program for young children that would help prepare them for the future they would be instrumental in shaping.

It was cultivated by Marilyn Buxton, whose deep interest in early childhood learning enabled her to find ways to teach primary children to program microcomputers.

It was fertilized by Devin Brown, with his lively wit and creative writing style. He gave it the nutrients it needed to appear in printed form to be shared. His shadow is cast over most of the later authors who patterned their style and examples after his original writings.

It was cared for by Howard Smith, Charles Miller, George Kolopanis, Alverta Darding, Lea Ann Hummel, Robin Koch and others, who worked with it in the lab helping to remove the bugs that would stunt its growth.

It was harvested by Nancy Taitt, Marilyn Hoots, Wayne Owens, Diane ZuHone, and others who typed and phoned and talked with people to spread the word and create a market for the final fruit.

And most important of all were the CHILDREN who tried and tested the materials that were produced. They shared their likes and dislikes, and made certain that everything that was included could be done by young minds.

These books were not created by a publisher to be sold to schools, where they would be used on children. They were instead, created from the successes of children, edited by the concerns of parents, and then offered to anyone that wishes to enrich the minds of young children.

If you elect to use these materials, then you assume the responsibility to encourage independent thought, reward creativity, enhance reasoning and logic, and above all, be forever open to alternate ways to solve problems.

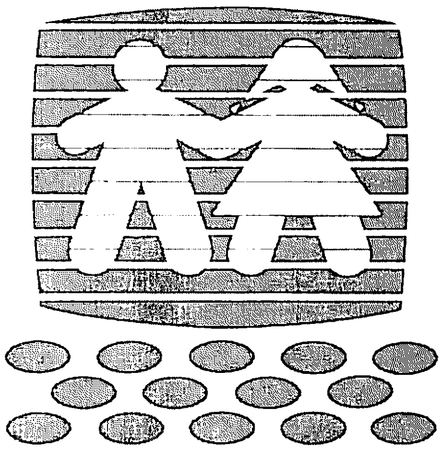
If you do this, your own rewards will be found in the faces of the children you serve.

Creative Programming for Young Minds

. . . on the TI-99/4A ™

Volume II

by Leonard Storm



© 1982, CREATIVE Programming, Inc., Charleston, IL 61920
A Subsidiary of R.V. Weatherford Co.

Hello, TI Level I Computer Programmer!

Are you ready to become a TI Level II Programmer?

Then, welcome to Volume II!

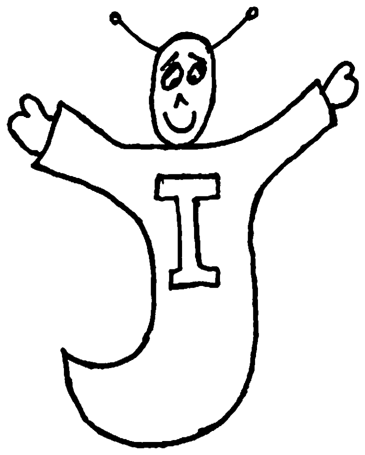
In Volume I, the following TI BASIC commands were discussed:

PRINT	LIST	CONTINUE	OLD
NEW	GOTO	CALL SCREEN	
RUN	END	BYE	
STOP	CALL CLEAR	RES	

Volume I also revealed the secrets of the following special keys:

$\boxed{\text{ENTER}}$	" (quotation mark)	SPACE BAR
$\boxed{\text{SHIFT}}$	$\boxed{\text{FCTN}}$	

The material in Volume II builds upon the concepts of Volume I. So if you have mastered the material covered earlier, then you are ready to continue with the next lesson, LESSON #5, brought to you by . . .



. . . TEX!

CREATIVE PROGRAMMING FOR YOUNG MINDS

...ON THE TI-99/4A

VOLUME II

TABLE OF CONTENTS

LESSON #5 FOR-NEXT

In this lesson, you are going to learn about some counting capabilities of the computer. So to get started, type in the following program:

```
10 CALL CLEAR
20 T=999
30 PRINT "T=";
40 PRINT T
```

Now, RUN the program.

You have already seen statements like 30 before. Statement 30 tells the computer to print everything between the quotation marks. But, statements 20 and 40 are new. Can you guess what these statements tell the computer to do? Statement 20 tells the computer that:

T equals 999. (= is an equals sign.)

But what does = mean?

T=999 means that T stands for 999 or that T has a number value of 999. So whenever the computer sees the letter T all by itself, it knows that you want the T to stand for the number 999.

Change statement 20 to this:

```
20 T=25
```

Then, RUN the program again. This time the computer thinks that T is the number 25.

Perhaps you have already guessed what statement 40 does. Instead of printing the letter T on the screen, it prints the number value of T.

Now type in this new program. Try to guess what it will print on the screen. Then check your guess by RUNning the program.

```
5 CALL CLEAR
10 NICKEL=5
20 DIME =10
30 PRINT "HOW MANY CENTS IN A NICKEL?"
40 PRINT NICKEL
50 PRINT "...IN A DIME?"
60 PRINT DIME
```

In the above program,

NICKEL has a value of _________

DIME has a value of _________

What would the following PRINT commands put on the screen?

PRINT NICKEL __________________

PRINT "NICKEL" _________________

Now everyone knows that 5 + 10 = 15. So one nickel plus one dime (or 5 cents plus 10 cents) equals 15 cents. If you tell the computer that NICKEL = 5 and DIME = 10, then NICKEL + DIME should equal (or stand for) 15.

Include the following program lines in your program:

```
70 PRINT "... IN A NICKEL AND A DIME?"
80 PRINT NICKEL + DIME
```

Now RUN the program and see what happens.

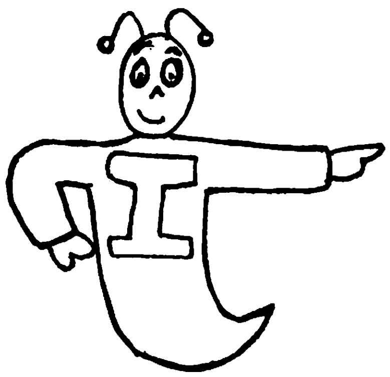
The value of NICKEL + DIME is 15 and statement 80 causes this value to be printed. The computer knows how to add!

Next, use the NEW command to erase the computer's memory or the program and then type the following program statements:

```
5 CALL CLEAR
10 PRINT T
20 R=1
30 PRINT R
40 R=2
50 PRINT R
60 R=3
70 PRINT R
```

Can you guess what the program will do? RUN the program to find out.

Names like T, NICKEL, DIME, and R which are given values may change or vary. They are called <u>variables</u>.



REMEMBER: PRINT "...LETTERS..." WILL PRINT EVERYTHING BETWEEN THE QUOTATION MARKS, WHILE PRINT VARIABLE WILL PRINT THE VALUE OF THE VARIABLE. IF THE VARIABLE HAS NOT BEEN GIVEN A VALUE, THEN THE COMPUTER WILL PRINT A ZERO.

Next, let's see how easy it is for the computer to count to 10. Type the following program into the computer's memory and then RUN the program.

```
100 FOR X=1 TO 10 <— This tells the computer the beginning
                     and ending numbers.
200 PRINT X <——————— This tells what to do.
300 NEXT X <———————— This says go to the next number.
```

The statement numbered 100 is a special kind of counting statement. It tells the computer that you want it to count from 1 up to 10. The statements in lines 100 and 300 form a loop.

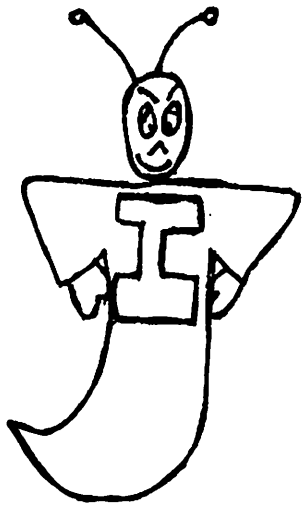
Change statement 100 to the following and then RUN the program.

```
100 FOR X=3 TO 20
```

This time the computer printed the whole numbers from 3 to 20.

What would the statement 100 have to be so that the computer counts from 50 to 200? Write your answer in the space below and then check your answer on the computer.

____



THE FOR STATEMENT TELLS THE COMPUTER AT WHAT VALUES TO START AND STOP COUNTING.

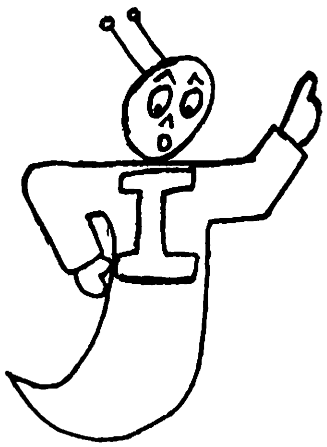
THE NEXT STATEMENT TELLS THE COMPUTER TO GIVE THE VARIABLE ITS NEXT VALUE, AND REPEAT THE LOOP IF IT HAS NOT YET LOOPED THE PRESCRIBED NUMBER OF TIMES.

As a further example, type in this new program.

```
5 PRINT "START"
10 FOR I=1 TO 3
15 PRINT "YODA LIVES!"
20 NEXT I
25 PRINT "STOP"
```

Try to figure out what the program will do. Then, check by RUNning the program.

The computer first prints START on the screen. Then it loops through the FOR-NEXT statements 3 times (for I=1, I=2, and I=3). Each time through, it prints YODA LIVES. Finally, the computer prints STOP on the screen.



FOR-NEXT ARE TWO SEPARATE STATEMENTS THAT WORK TOGETHER. ONE IS NO GOOD WITHOUT THE OTHER.

FOR-NEXT BOTH USE THE SAME VARIABLE. THIS VARIABLE CAN BE ALMOST ANY COMBINATION OF LETTERS YOU WANT IT TO BE.

FOR-NEXT IS A QUICK WAY TO GET THE COMPUTER TO DO SOMETHING A CERTAIN NUMBER OF TIMES.

Now type in this example and RUN it.

```
5 CALL CLEAR
10 FOR TEX=1 TO 5
20 PRINT "  *"
30 PRINT " ***"
40 PRINT "*****"
50 PRINT "  *"
60 PRINT "  *"
70 PRINT "  *"
80 PRINT
90 NEXT TEX
```

The arrow is printed on the screen 5 times, for TEX=1, 2, 3, 4, and 5.

How many times would the arrow be printed on the screen if statement 10 were:

10 FOR TEX=3 TO 7 ? ____________________

Change statement 10 and find out.

Now erase statement 10.

Type in these additional program statements.

```
90 FOR TEX=1 TO 24
100 PRINT
110 NEXT TEX
```

Now LIST the program. Statement 10 should be gone. Statements 20 to 80 should be the same as before and statements 90 to 110 should be the ones you've just typed in.

Guess what the program will do, then run the program to find out for sure.

Statements 90, 100, and 110 cause a ______________________ to be printed __________ times. This causes the arrow to scroll upward.

Are you starting to understand this FOR-NEXT business? Great! Then it's time for some practice exercises.

EXERCISE 5-1

Write a program (using FOR and NEXT statements) that will cause your computer to print the numbers from 1 to 1000. Use your first name as the FOR-NEXT variable.

Write your program on the lines below and then RUN it. (Try timing the program to see how long the computer takes to count to 1000.)

After you get your program to RUN, change it so that the computer counts up to some different number. Write the change that you made on the line below.

Now change the program so that it will cause your name to be printed out each time it prints out a number. For example:

```
1
your name
2
your name
3
your name
```

Only one extra line is needed. Write it on the line below. Then RUN your program.

EXERCISE 5-2

FOR-NEXT loops are sometimes used as delays. The following programs illustrate how delays can be useful.

Type the following program into the computer and RUN it.

```
10 CALL SCREEN(7)
20 PRINT "THE SCREEN IS RED."
```

The screen didn't stay red very long, did it?

In Volume I, you've already seen one way to keep the screen red by using an infinite loop. Type in the line

```
30 GOTO 30
```

and RUN the program again.

This keeps the screen red, but doesn't allow anything new to occur.

Now type in the following statements:

```
30 FOR DELAY=1 TO 500
40 NEXT DELAY
50 PRINT "TIME'S UP."
```

LIST the program to see what statements are now in the program and then RUN the program. Do you see how the FOR-NEXT statement provides a time delay? After this short delay, the program can move on to statement 50.

Now put this new program into the computer.

```
5 CALL CLEAR

10 PRINT "START TIMING"

20 FOR X=1 TO 1000

30 NEXT X

40 PRINT "STOP TIMING"
```

RUN this program and try to time how long the delay lasts. Then change statement 20 so that the computer counts to 2000, then 3000, etc. Time the program each time and fill in the following blanks.

If the computer counts from 1 to:	it will take this number of seconds:
1000	____________
2000	____________
3000	____________
3500	____________
5000	____________
10000	____________

EXERCISE 5-3

Remember that FOR and NEXT statements are always used together. If one has been left out of a program, the computer will give you an error message.
Find out what error messages are printed out by the computer when you RUN the following programs.

PROGRAM #1

```
10 FOR I=1 TO 100
20 PRINT I
```

ERROR MESSAGE: ______________________________

PROGRAM #2

```
10 I=6
20 PRINT I
30 NEXT I
```

ERROR MESSAGE: ______________________________

PROGRAM #3

```
10 FOR TI=1 TO 100
20 PRINT TI
30 NEXT I
```

ERROR MESSAGE: ______________________________

Explain what was wrong in each program.

PROGRAM #1: ______________________________

PROGRAM #2: ______________________________

PROGRAM #3: ______________________________

EXERCISE 5-4

All of the FOR-NEXT statements that you have studied so far count by ones. Wouldn't it be handy if you could get the computer to count by two's (2, 4, 6, 8, ...), or by fives (5, 10, 15, 20, ...), or by some other number?

The following program illustrates counting by two's. Type the program into the computer and RUN it.

```
5 CALL CLEAR
10 FOR J=0 TO 10 STEP 2
20 PRINT J
30 NEXT J
```

In the above program, J starts with a value of 0. The STEP 2 part of statement 10 causes the value of J to increase by two every time the program reaches statement 30. The last printed value of J is 10.

Now add the following statement to the program.

```
40 PRINT J
```

Can you guess what statement 40 will print out? RUN the program to find out. What was printed? ______________

Remember that statement 30 increases J by two every time it is executed. On the last time through the loop, statement 20 prints a 10 on the screen. Then statement 30 adds two to J. Now J=12, which is greater than the limit 10, so the program quits looping. Statement 40 then prints the last value of J which is still 12.

Now it's up to you! Write a program that will print out the numbers from 3 to 21 by three's (3, 6, 9, ...). Write your program below.

Type the program into the computer and RUN it. When the program is working properly, then you may proceed.

The computer can also be made to count backwards. In a FOR-NEXT loop, this is done by using a negative value for the STEP. For example,

```
FOR I=10 TO 4 STEP -2
```

tells the computer to start at 10, and count backwards to 4 by two's.

Now write a program that will print the numbers from 10 down to 1 by ones. After this, the program should print out BLAST OFF! Write the program on the lines below and then check the program by RUNning it. When it works properly, go to the next exercise.

EXERCISE 5-5

Here is another BLAST OFF program which uses a FOR-NEXT loop inside a FOR-NEXT loop. Type this program into the computer and RUN it.

```
100 CALL CLEAR
110 FOR COUNTDOWN=10 TO 1 STEP -1
120 PRINT COUNTDOWN
130 FOR DELAY=1 TO 500 }  This delay allows the screen
140 NEXT DELAY         }  to be read.
150 CALL CLEAR
160 NEXT COUNTDOWN
170 PRINT "BLAST OFF!"
```

This is how the program works: First, the screen clears. Next, the value of COUNTDOWN is set to 10. Statement 120 then prints this value. Statements 130 and 140 provide a short delay before statement 150 clears the screen. Next, statement 160 sets COUNTDOWN equal to 9. Again, statement 120 prints the value of COUNTDOWN. After a short delay, the 9 is erased and so on.

Wow! Those few statements do a lot of work.

What will happen if you change statement 130 to:

```
130 FOR DELAY=1 TO 10
```

Make the change and RUN the program to find out.

Next, add this program statement.

```
135 PRINT " ."
```

Also, get rid of statement 150 and then LIST the program.

Can you guess what the computer will print on the screen this time? RUN the program and fill in the spaces below.

The program prints _____, then _____ dots, a _____, then _____ dots, an _____, then _____ dots, and so forth down to _____. Finally, the computer prints BLAST OFF!

Each time, the 10 dots come from the inside FOR-NEXT loop which loops 10 times through a PRINT " ." statement. The printed numbers come from the outside FOR-NEXT loop.

Now it's your turn. Write a program which contains one FOR-NEXT loop inside another FOR-NEXT loop. Try to get the computer to print out the following cheer:

```
YEA, TEX!
TEX!
TEX!
TEX!
YEA, TEX!
TEX!
TEX!
TEX!
YEA, TEX!
TEX!
TEX!
TEX!
```

HINT: Use the outside FOR-NEXT loop to print YEA, TEX! three times and use the inside FOR-NEXT loop to print TEX! three times.

Be sure to use different variables for the two different loops.

You may use the lines on the next page to write your program.

What line of your program would you have to change to make your program print YEA, TEX! ten times? Write the change below and then try it out on the computer.

Now add another FOR-NEXT loop to your program. This one should serve as a delay to slow down the printing speed as shown below:

```
YEA, TEX!  ← Pause after YEA, TEX!
TEX!
TEX!
TEX!
YEA, TEX!  ← Pause after YEA, TEX!
TEX!
TEX!
TEX!
(Etc.)
```

Can you figure out where the new FOR-NEXT loop should go in your program? Try it!

EXERCISE 5-6

Now you're on your own. Write a program which uses FOR-NEXT statements which cause something to be done a certain number of times. You should also try to use STEP, GOTO, PRINT, CALL CLEAR, and CALL SCREEN.

EXERCISE 5-7

In this exercise, you will learn how the limits on a FOR statement can be changed without typing in a new FOR statement.

Enter the following program into your computer. See if you can figure out what it will do, then RUN the program.

```
10 FOR LIMIT=1 TO 5
20 PRINT "NOW LIMIT="
30 PRINT LIMIT
40 PRINT
50 FOR DELAY=1 TO 500
60 NEXT DELAY
70 FOR COUNT=1 TO LIMIT
80 PRINT COUNT
90 NEXT COUNT
100 PRINT
110 NEXT LIMIT
```

Statement 70 contains a changing or variable limit which can be easily changed. In this program, three FOR-NEXT loops are used. The outer one (from statements 10 to 110) sets the value of LIMIT. Statements 20 and 30 tell what the value of LIMIT is. Statements 50 and 60 form a small time delay loop. Finally, statements 70, 80, and 90 form a counting loop which prints the value of COUNT from 1 up to the value of LIMIT as set by the outer FOR-NEXT loop.

Look over this next program, but don't put it into your computer yet. First answer the questions following it, then check your answers by RUNning the program.

```
10 A=50
20 B=10
30 C=115
40 FOR COUNT=A TO C STEP B
50 PRINT COUNT
60 NEXT COUNT
```

What will be the first number printed on the screen? ________

What will be the last number printed on the screen? _________

The computer will count by _________ (one's, two's, etc.).

Change the program so that the computer counts from 10 to -10 by a step of -2. Now,

A= __________

B= __________

C= __________

You've learned a lot already! But when you're ready, there are even more good things to come. So for more computing adventures, go to LESSON #6.

LESSON #6 COMMA (,) SEMI-COLON (;) AND COLON (:)

This lesson will show you some new PRINT tricks using:

the comma (,)
the semi-colon (;) and
the colon (:) .

These symbols are known as PRINT separators because they are used to separate items in a PRINT list.

Let's first see how the comma is used in a PRINT statement. Enter the following program into the computer and then RUN it.

```
10 DIGIT=33
20 PRINT "DIGIT=", DIGIT
30 PRINT
40 PRINT "LEFT ZONE", "RIGHT ZONE"
50 PRINT
60 PRINT 1, 2, 3, 4, 5, 6, 7
```

Notice that the computer has printed everything in one of two zones on the TV screen. The left zone begins in the first column on the left of the screen. The right zone begins fifteen columns over.

Using the comma, one may put more than one item in a PRINT statement. The comma causes each item in the PRINT statement to be printed in a different zone.

To check whether you understand how the comma works, use the following program to fill in the blanks below it.

```
10 PRINT "X=", "Y="
20 X=6
30 PRINT X,Y
40 X=10
50 Y=3
60 PRINT X,Y,X,8+X
```

Write down what you think the computer will print in each zone.

LINE	LEFT ZONE	RIGHT ZONE
1		
2		
3		
4		

RUN the program to check your answers.

The next program illustrates the use of the semi-colon. Type it into the computer and watch it RUN.

```
10 X=256
20 Y=128
30 PRINT "X="; X
40 PRINT "Y=", Y
```

This program illustrates the difference between a comma and a semi-colon in a PRINT statement.

The comma causes items to be printed in separate zones. The semi-colon causes items to be printed one right after another.

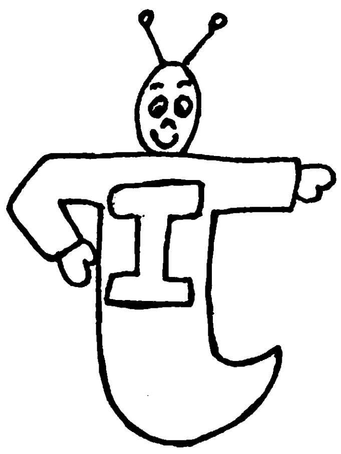
Type in the next program and RUN it.

```
10 PRINT "WATCH"; "OUT"
20 PRINT "WATCH "; "OUT"
```

Notice that the semi-colon (;) causes an item to be printed where the last print statement ends. For this reason, statement 10 caused the words WATCH and OUT to be run together. In statement 20, an extra space was put after the first word so that WATCH and OUT would be separated by one space.

Examine the next program. Try to figure out what it will do before you RUN it. Then type the program into the computer and RUN it.

```
5 CALL CLEAR
10 PRINT "WATCH ME COUNT TO 10"
20 FOR I=1 TO 10
30 FOR J=1 TO 500
40 NEXT J
50 PRINT I; " COMES NEXT"
60 NEXT I
70 PRINT "I'M DONE!"
```



THERE ARE THREE KINDS OF PRINT SEPARATORS:

THE COMMA (,) CAUSES ITEMS TO BE PRINTED IN DIFFERENT ZONES.

THE SEMI-COLON (;) CAUSES ITEMS TO BE PRINTED ONE AFTER ANOTHER.

THE COLON (:) CAUSES THE NEXT ITEM TO BE PRINTED ON A NEW LINE.

The following program illustrates the use of the colon (:) in a PRINT statement. Enter the program into the computer and RUN it.

```
10 CALL CLEAR
20 PRINT "MY NAME IS TEX.": "WHAT'S YOURS?"
30 GOTO 30
```

Notice that the colon caused WHAT'S YOURS? to be printed on a new line.

Print separators can also be used one after another with nothing in between. RUN the following program to see how easy it is to scroll by 5 lines.

```
10 CALL CLEAR
20 PRINT "SCROLL UPWARD BY 5 LINES!":"O.K.":::::"DONE!"
30 GOTO 30
```

Wow! All of that in one program line!

Now let's look at a nifty little program that uses some of the new things you've learned. The program tells the number of days in a certain number of years.

```
10 CALL CLEAR
20 PRINT "YEARS", "DAYS"
30 FOR I=1 TO 10
40 PRINT I, 365*I:
50 NEXT I
60 GOTO 60
```

THE * SYMBOL IS THE COMPUTER'S MULTIPLICATION SIGN. IT TELLS THE COMPUTER TO MULTIPLY TWO NUMBERS TOGETHER. FOR EXAMPLE 2*6=12.

Can you guess what the following program does? Try it and see.

```
10 CALL CLEAR
20 FOR I=100 TO 1 STEP -1
30 PRINT I;
40 NEXT I
50 GOTO 50
```

Each time statement 30 is encountered, the semi-colon causes the new value of I to be printed right after the last value.

EXERCISE 6-1

Write a program that will print HERE ARE 500 A'S, then skip 5 lines, and finally, print out 500 A's one right after the other. In your program, use the colon and semi-colon separators. Show your working program on the lines below.

Write a program that will print numbers by five's from five to one hundred. Have all the numbers ending in 5 printed in the first zone, and all the numbers ending with zero in the second zone. The screen should look like this:

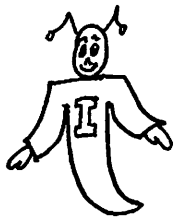
```
         COLUMN 1                COLUMN 2

         5                       10
         15                      20
         25                      30
         35                      40
         45                      50
         55                      60
     (and so on)
```

Use a comma separator to do the job.

LESSON #7 GRAPHICS

In an earlier lesson, you learned how to change the color of the TV screen by using the SCREEN command. In this lesson, you are going to find out that the TI computer can do a lot of things with color. For instance, the computer can make pictures of moving objects such as cars, robots, inchworms, dragons, and almost anything. Are you ready to learn how to do these things? Then read on!!



THE PICTURES WHICH YOU WILL LEARN TO MAKE ARE CALLED GRAPHICS. GRAPHICS ARE FUN!

To begin, type in the following program. Then, RUN it. Don't worry about what the statements do. You'll learn about that later.

```
10 CALL CLEAR
20 CALL SCREEN (2)
30 CALL CHAR (96,"0066FFFFFF7E3C18")
40 CALL COLOR (9,7,12)
50 FOR V=2 TO 24 STEP 2
60 FOR H=4 TO 28 STEP 2
70 CALL VCHAR(V,H,96)
80 NEXT H
90 NEXT V
100 GOTO 100
```

These are zeroes.

(Make sure the letters are in uppercase.)

Hearts anyone?!

Now type in the following program and RUN it.

```
10 CALL CLEAR
20 CALL COLOR(2,7,16)
30 CALL HCHAR(5,5,42)
100 GOTO 100
```

The above program puts an asterisk (*) on the screen at a certain position. The position is given in statement number 30. To see that this is so, add the program line below and then RUN the program again.

```
31 CALL HCHAR(6,5,42)
```

Do you see that the only difference between statement 30 and statement 31 is the number 6?

Add the next statement to the program and see what it does.

```
32 CALL HCHAR(8,5,42)
```

Notice that the higher the first number is in the HCHAR command, the lower the character occurs on the screen. On the line below, write an additional statement for the above program. This statement should cause an asterisk to be printed in the missing spot.

33 ____

RUN the program to make sure that the new statement does its job. Try adding other statements which print an asterisk in different screen positions.

Try to find out the highest and lowest number that can be used in the HCHAR command for the first number. Record your findings on the blanks below.

```
HIGHEST VALUE:  CALL HCHAR (__________,5,42)
LOWEST VALUE:   CALL HCHAR (__________,5,42)
```

Now, let's experiment with the second number in the HCHAR command. RUN the following program.

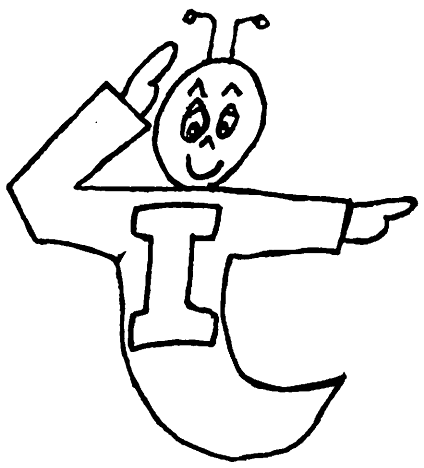
```
10 CALL CLEAR
20 CALL COLOR(2,7,12)
30 CALL HCHAR(5,5,42)
40 CALL HCHAR(5,6,42)
50 CALL HCHAR(5,10,42)
100 GOTO 100
```

The second number in the HCHAR command tells the computer how far to the right to print the character.
Add some program lines of your own. Try to find out the highest and lowest number which can be put in the second position of an HCHAR command.

```
HIGHEST VALUE:  CALL HCHAR (5,__________,42)
LOWEST VALUE:   CALL HCHAR (5,__________,42)
```

Try to find out what HCHAR command would print an asterisk in the middle of the TV screen. Then write the statement on the line below.

____



HCHAR STANDS FOR <u>H</u>ORIZONTAL <u>CHAR</u>ACTER. IN AN HCHAR STATEMENT, THE FIRST NUMBER TELLS HOW FAR DOWN FROM THE TOP OF THE SCREEN TO PRINT THE CHARACTER.

THE SECOND NUMBER TELLS HOW FAR TO THE RIGHT TO PRINT THE CHARACTER.

What statement would print an asterisk 18 lines down from the top of the screen and 30 spaces from the left of the screen? ____________________

The third number of the HCHAR command tells the computer what character to print. The number 42 is the code for * .

If you're curious, you may want to add some other program statements to the last program, this time changing the 42 to some other number. For instance, what does a 43 represent?

Now let's investigate the CALL COLOR command. This command tells the computer what color to make a character. Notice that in the last two programs, the CALL COLOR command comes before the CALL HCHAR command. It has to! The color of the character must be stated before the character is printed by the HCHAR statement!

The first number in the COLOR command tells the SET or group of the character you want to print. In the examples given above notice that character 42, an asterisk, is in set 2.

The next number in the COLOR command, the middle number, tells what color to make the character. In the last example, this number was a 7 which is the code for dark red. (Remember, the color code was given in Volume I.)

The last number in the COLOR command tells what color to make the square behind the character. In the last example, this number was a 12 which represents light yellow.

Below you will find a list of all the characters in set 2.

SET NUMBER TWO CHARACTER CODES

(	40	left parenthesis
)	41	right parenthesis
*	42	asterisk
+	43	plus
,	44	comma
-	45	minus
.	46	period
/	47	slash

Since there is quite a bit to remember about graphics, let's practice! Practice makes perfect!

Suppose you want to print a black slash on a dark red square. The color command would be:

```
CALL COLOR(2,2,7)
           ↑ ↑ ↑
           | | └──dark red square or background
           | └──black character
           └──slash character is in set 2
```

Now to print the slash, you would have to use the HCHAR command:

```
CALL HCHAR(18,24,47)
            ↑  ↑  ↑
            |  |  └──code for slash
            |  └──print 24 spaces from the left
            └──print 18 rows down from the top
```

Try to answer the following questions by filling in the blanks with the right numbers.

1. To make the color of a set 2 character be dark blue on a light blue square, what statement would you use?

   CALL COLOR(______,5,______)

2. To make the color of a set 4 character be dark blue on a light blue square, what statement would you use?

   ```
   CALL COLOR(______,5,6)
               ↑
               └──Not 2 this time!
   ```

3. To make the color of a set 2 character be white on a dark red square, what statement would you use?

   ____

NOTE: A CALL COLOR command causes all the characters in a certain set to have the same color. For example: CALL COLOR(2,7,16) would tell the computer that all set 2 characters are to be printed in dark red on a white background.

Try to guess what the following program will do. Then RUN the program.

```
10 CALL CLEAR
20 CALL COLOR(2,7,7)
30 CALL HCHAR(4,4,47)
40 CALL HCHAR(4,5,47)
50 CALL HCHAR(4,6,47)
60 CALL HCHAR(4,7,47)
70 CALL HCHAR(4,8,40)
80 CALL HCHAR(4,9,40)
90 CALL HCHAR(4,10,40)
100 CALL HCHAR(4,11,40)
110 GOTO 110
```

Notice lines 30 through 60 print a slash and lines 70 through 100 print a left parenthesis: <u>47 stands for /</u> and <u>40 stands for (</u> .

But why did all the characters look like dark red squares? The reason is that the character color, (7), is the same as the background color, (7). See statement 20. Both characters 40 and 47 are given the same colors because both are in the same set, (2).

The previous program shows one way to draw a line using squares. But happily, there is an even easier way. RUN the following program to see how.

```
10 CALL CLEAR
20 CALL COLOR(2,2,2)
30 CALL HCHAR(4,4,47,10)
40 GOTO 40
```

Statement 20 causes set 2 characters to be printed in black on a black background. Statement 30 actually does the printing. It tells the computer to print character 47 (black on black) starting at position 4 down and 4 over. The last number tells the computer to print 10 of these characters side-by-side in a horizontal row.

Now RUN the following program.

```
10 CALL CLEAR
20 CALL COLOR(2,7,7)
30 CALL HCHAR(4,4,42,10)
40 CALL VCHAR(4,14,42,10)
50 CALL HCHAR(14,4,42,11)
60 CALL VCHAR(4,4,42,10)
70 GOTO 70
```

The VCHAR command is very similar to the HCHAR command. VCHAR stands for <u>V</u>ertical <u>CHAR</u>acter.

```
CALL VCHAR(4,14,42,10)
           ↑  ↑  ↑  ↑
           │  │  │  └── print 10 characters in a vertical column
           │  │  └───── print character 42
           │  └──────── print the first character 14 spaces from the left
           └─────────── print the first character 4 rows down from the top
```

The only difference between HCHAR and VCHAR is that HCHAR prints to the right while VCHAR prints downward.

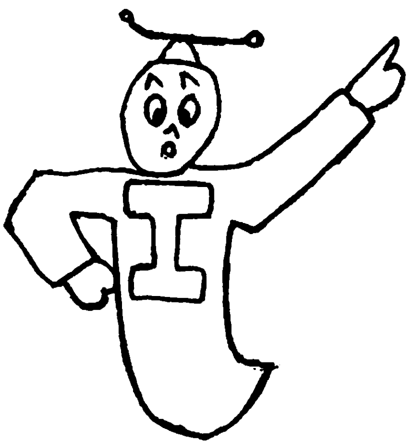
Now add a statement to this program that will print character 42 in row 4 and column 9 and that will print 10 more underneath this position:

65 ______________________________

Statement 65 should cause a vertical line to be printed from the top to bottom across the center of the square.

Now write another statement that will cause a horizontal line to be printed from one side of the square to the other through the center of the square.

66 ______________________________



REMEMBER: FIRST DEFINE THE COLOR FOR A SET USING CALL COLOR. THEN YOU MAY USE HCHAR TO PRINT HORIZONTAL ROWS OF CHARACTERS IN THE SET OR VCHAR TO PRINT VERTICAL COLUMNS OF CHARACTERS IN THE SET.

WHEN YOU GET READY TO RUN THE NEXT COLOR PROGRAM, FIRST SALUTE . . . YOU'LL SEE WHY!

```
10 CALL CLEAR             ----------> clear the screen

20 CALL SCREEN(2)         ----------> blacken the screen

30 CALL COLOR(2,16,5)     ----------> set 2 is white on blue

40 CALL COLOR(3,7,7)      ----------> set 3 is red on red

50 CALL COLOR(4,16,16)    ----------> set 4 is white on white

60 CALL COLOR(5,5,5)      ----------> set 5 is blue on blue

70 FOR ROW=6 TO 18                 }
                                   }  makes a red area
80 CALL HCHAR(ROW,4,48,25)         }  character 48 is in set 3
                                   }
90 NEXT ROW                        }

100 FOR ROW=8 TO 17 STEP 2         }
                                   }  makes white stripes
110 CALL HCHAR(ROW,4,56,25)        }  character 56 is in set 4
                                   }
120 NEXT ROW                       }

130 FOR ROW=6 TO 12                }
                                   }  makes a blue field
140 CALL HCHAR(ROW,4,64,11)        }  character 64 is in set 5
                                   }
150 NEXT ROW                       }

160 FOR ROW=6 TO 12 STEP 2         }
                                   }
170 FOR COLUMN=4 TO 14 STEP 2      }
                                   }  puts 6 stars in a line
180 CALL HCHAR(ROW,COLUMN,42)      }  character 42 is in set 2
                                   }
190 NEXT COLUMN                    }
                                   }
200 NEXT ROW                       }

210 FOR ROW=7 TO 11 STEP 2         }
                                   }
220 FOR COLUMN=5 TO 13 STEP 2      }
                                   }
230 CALL HCHAR(ROW,COLUMN,42)      }  puts 5 stars on a line
                                   }
240 NEXT COLUMN                    }
                                   }
250 NEXT ROW                       }

260 GOTO 260              ----------> infinite loop
```

In this program, statements 30 to 60 set the colors of the various sets. Next, statements 70 through 90 print out a red rectangle using the HCHAR command in statement 80. The HCHAR statement causes character 48 to be printed 25 times in a horizontal line beginning at row ROW and column 4. Since character 48 is in set 3 which has been given the color of red on red, then printing the character 48 will print a red square. The FOR statement in line 70 causes the value of ROW to change so that several rows of red lines are produced.

Now look at statements 100 to 120. What character does line 110 print? ___________

You are told that this character is in set 4.

What color is plotted when the 56 is plotted? _____________

In what row and column is the first 56 character plotted?

____

How many times is character 56 plotted in one line? _________

Now take some time to see if you can understand how the rest of the program works. Especially, try to understand how the FOR-NEXT loops work. Statements 160 through 200 contain one FOR-NEXT loop inside another. That is, one FOR-NEXT loop is nested inside another. You saw this a little earlier in this manual.

To refresh your mind on FOR-NEXT loops (in case you've forgotten), let's see how steps 160 through 200 are executed:

```
160  ROW is set to 6.
170  COLUMN is set to 4.
180  A star is printed on row 6, column 4.
190  COLUMN is set to 6.
180  A star is printed on row 6, column 6.
190  COLUMN is set to 8.
180  A star is printed on row 6, column 8.
190  COLUMN is set to 10.
180  A star is printed on row 6, column 10.
190  COLUMN is set to 12.
180  A star is printed on row 6, column 12.
190  COLUMN is set to 14.
180  A star is printed on row 6, column 14.
190  COLUMN=16 is beyond the limit.
200  ROW is set to 8.
170  COLUMN is set to 4 again.
and so on . . .
```

In other words, the inside FOR-NEXT loop prints a row of 6 stars with a space between each star. The outer FOR-NEXT loop changes the row number. Then the inside FOR-NEXT loop prints 6 stars on this new row, etc.

Now try this program.

```
10 CALL CLEAR
20 CALL COLOR(2,2,2)
30 CALL HCHAR(8,20,42)
40 FOR I=1 TO 1000
50 NEXT I
60 CALL COLOR(2,7,12)
70 GOTO 70
```

Notice that in statement 20 the set 2 is assigned a color of black for both the characters and the background. Then statement 30 prints a black asterisk on a black square. Later, statement 60 changes the color code for set 2. What happens when statement 60 is executed? ____________________

____

As a final example, RUN the following program.

```
10 CALL CLEAR
20 CALL COLOR (2,2,7)
30 CALL HCHAR(5,5,42,40)
40 CALL VCHAR(2,5,43,40)
50 GOTO 50
```

Notice what happens when the HCHAR or VCHAR statements try to print more characters than are in a row or column. It continues on the next line or column.

Try changing statement 40 so that the screen is filled with the character whose code is 43.

40 ____

EXERCISE 7-1

For your convenience a TABLE OF CHARACTER CODES has been included on page 85. Use the table to fill in the following blanks. See Volume I for the color codes.

What numbers are needed in the following statements so that a black 9 is printed on a white square at a position 5 rows down from the top of the screen and 10 spaces from the left?

```
     10 CALL COLOR(_______,_______,_______)
     20 CALL HCHAR(_______,_______,_______)
     30 GOTO 30
```

RUN this short program to check your answer.

What numbers are needed in the following statements so that a yellow square is printed at position (row 10, column 10) and a black square is printed at position (row 20, column 20)?

```
     10 CALL COLOR(_______,_______,_______)
     20 CALL HCHAR(_______,_______,_______)
     30 CALL COLOR(_______,_______,_______)
     40 CALL HCHAR(_______,_______,_______)
     50 GOTO 50
```

Again, RUN the program to check your answers.

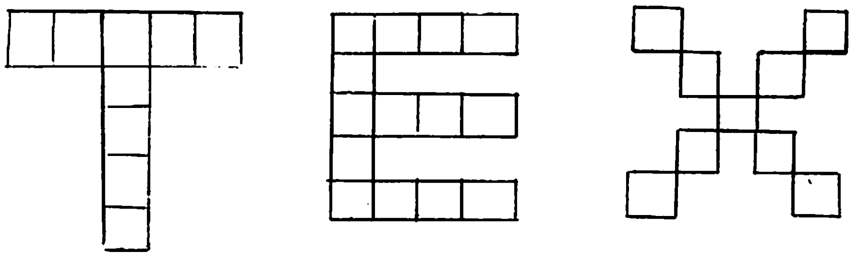
HINT: Each color needs a different set number. Don't use set 1 since it contains the space character. The screen is filled with spaces!

TABLE OF CHARACTER CODES

SET	CHARACTER	CODE	SET	CHARACTER	CODE
1	(space)	32	5	@	64
1	!	33	5	A	65
1	"	34	5	B	66
1	#	35	5	C	67
1	$	36	5	D	68
1	%	37	5	E	69
1	&	38	5	F	70
1	'	39	5	G	71
2	(	40	6	H	72
2	)	41	6	I	73
2	*	42	6	J	74
2	+	43	6	K	75
2	,	44	6	L	76
2	-	45	6	M	77
2	.	46	6	N	78
2	/	47	6	O	79
3	0	48	7	P	80
3	1	49	7	Q	81
3	2	50	7	R	82
3	3	51	7	S	83
3	4	52	7	T	84
3	5	53	7	U	85
3	6	54	7	V	86
3	7	55	7	W	87
4	8	56	8	X	88
4	9	57	8	Y	89
4	:	58	8	Z	90
4	;	59	8	[	91
4	<	60	8	\	92
4	=	61	8	]	93
4	>	62	8	^	94
4	?	63	8	_	95

EXERCISE 7-2

Write a program using HCHAR and VCHAR which will cause a large TEX to be printed on the screen. Make the letters out of red squares. Let the screen color be black.



EXERCISE 7-3

Now write a program that will make a picture of a tree. Be creative! For instance, you might want to include some fruit on your tree.

You will find a graphics chart on page 88. It shows the rows and columns of all the squares on the screen. You may use it to help design your tree. Just shade in the squares that you want to light up on the screen to form the tree.

When you get the program to RUN as you want it to, DON'T ERASE IT! YOU WILL USE THIS PROGRAM IN THE NEXT LESSON.

TI GRAPHIC SHEET

COLUMNS

ROWS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1																																
2																																
3																																
4																																
5																																
6																																
7																																
8																																
9																																
10																																
11																																
12																																
13																																
14																																
15																																
16																																
17																																
18																																
19																																
20																																
21																																
22																																
23																																
24																																

LESSON #8 SAVE

Sometimes when you write a long program, you will want to save it on cassette tape. By saving the program on tape, you can save yourself a lot of time. You won't have to type the long program into memory every time you want to run the program. This lesson will show you how to save a program on tape. Follow the steps below to record your last program.

HOW TO SAVE A PROGRAM ON TAPE

1. First type your program into the computer's memory. When you are satisfied that it RUNs perfectly, then you are ready to save your program.

2. To save the program, type:

   ```
   SAVE CS1
   ```
   (This must be written in uppercase.)

   and press $\boxed{\text{ENTER}}$. From now on the computer will guide you through the proper steps.

3. After step 2, the screen will read:

   ```
   *  REWIND CASSETTE TAPE    CS1
      THEN PRESS RETURN
   ```

   If you haven't put a blank cassette in the recorder yet, do it now. Then rewind the cassette tape by pushing the rewind button on the recorder. When the tape is completely rewound, press the stop button on the recorder. Then press $\boxed{\text{ENTER}}$ on the computer.

4. The computer will respond with:

```
*  PRESS CASSETTE RECORD    CS1
   THEN PRESS ENTER
```

Do it. Press the play and record buttons at the same time. This will get the recorder ready. Then press $\boxed{\text{ENTER}}$ on the computer.

5. The computer will then print out this message:

```
*  RECORDING
```

You should be able to hear the sound of the program information being stored.

Finally, the computer will print:

```
*  PRESS CASSETTE STOP    CS1
   THEN PRESS ENTER
```

You respond by pressing the stop button on the recorder. Then press $\boxed{\text{ENTER}}$ on the computer.

6. Your program has been recorded. Now the computer will ask if you want the tape to be checked to see if the recorder did its job properly. It will say:

```
*  CHECK TAPE (Y OR N)?
```

It is a good idea to check your tape. So press Y. (It must be capital Y.)

7. The computer will respond with:

```
*  REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER
```

Just do as the computer directs. Rewind your tape to

Keep going. $\longrightarrow$

the point where you began recording the program. Press stop when the tape has reached this position. Then press $\boxed{\text{ENTER}}$ on the computer.

8. The computer will respond with:

```
* PRESS CASSETTE PLAY    CS1
  THEN PRESS ENTER
```

Press the play button on the recorder. Then press $\boxed{\text{ENTER}}$ on the computer.

9. The computer will print:

```
* CHECKING
```

If there are no errors, the following messages will be printed on the screen:

```
* DATA OK

* PRESS CASSETTE STOP    CS1
  THEN PRESS ENTER
```

Do as the computer directs. You can then remove your tape. Now skip ahead to step 11.

10. If the program was not properly recorded, an error message will be displayed. Or, if for some reason the recorder did not play back properly, an error message will result.

But again, the computer will direct you. It will display:

```
PRESS R TO RECORD CS1
PRESS C TO CHECK
PRESS E TO EXIT
```

Before pressing R or C to try recording or checking

Keep going. ⟶

again do the following:

*Make sure the TV is 2 feet or more from the recorder.

*Make sure the recorder is attached to the computer correctly. See Volume I OLD CS1 instructions.

*Is the cassette tape in good condition? If you don't know for sure, try another tape.

*Is the cassette recorder volume set properly?

*Does the recorder tape head need cleaning?

*Is the system on a metal surface? It shouldn't be.

Now press R if you want to try recording again. Or press C if you want to check again. Then, just follow the computer directions as before.

11. Since you have successfully recorded your program, let's see if you can load it. First, clear computer memory (NEW). Now use OLD CS1 to load your program. Then RUN your program.

12. To be safe, you should record your program on two different tapes. That way, if something awful happens to one, you still have a back-up.

____
Name

VOLUME II REVIEW QUIZ

Fill in the blank with the correct word from the ANSWER POOL. (If you get stuck, turn back to the correct page and review.)

ANSWER POOL

variable	FOR X=0 TO 10 STEP 2	*
CALL VCHAR(10,20,58,3)	1	PRINT T
: (colon)	graphics chart	15
nested loop	SAVE CS1	=
CALL COLOR(3,7,2)	FOR-NEXT	0 (zero)
; (semi-colon)	, (comma)	graphics
58	CALL HCHAR(10,20,58,3)	

1. ____________________ is a symbol which means "has the same value" or "equals" (p.45).

2. ____________________ is a PRINT command that would print the value of T (p.45).

3. ____________________ is the name given to symbols which are given values which may vary (p.47).

4. ____________________ is the value of a variable if you haven't given the variable a value (p.47).

5. ____________________ is a pair of commands which causes the computer to do something a certain number of times (p.50).

6. ____________________ is a FOR statement that would count from 0 to 10 by two's (p.56).

7. __________________ is a PRINT separator which causes characters to be printed in different zones (p.64).

8. __________________ is the column in which zone one begins (p.64).

9. __________________ is the column in which zone two begins (p.64).

10. ________________ is a PRINT separator which causes characters to be printed one right after another (p.66).

11. ________________ is a PRINT separator which causes the next item to be printed on a new line (p.67).

12. ________________ is a symbol which causes the computer to multiply two numbers (p.68).

13. ________________ is the name for the picture drawing capabilities of your computer (p.71).

14. ________________ is a command that tells the computer that set 3 characters will be printed dark red on black (p.76).

15. ________________ is the code number which represents a colon in set 4 (p.85).

16. ________________ is a command that tells the computer to print 3 colon symbols in a horizontal row beginning at row 10 and column 20 (p.78).

17. ___________________ is a command that tells the computer to print 3 colon symbols in a vertical column beginning at row 10 and column 20 (p.79).

18. ___________________ is a term used for one FOR-NEXT loop which is completely inside another FOR-NEXT loop (p.81).

19. ___________________ is a chart which can be used to help design pictures for the computer (p.87).

20. ___________________ is the command which is used to save a program on cassette one (p.89).

THE COLORED PAGES

At the end of each manual, you will find several colored pages. These are projects that test your ability to use what you have learned. There are no right or wrong answers. If your program does what is asked, then it is quite acceptable. You are free to express your creativity. Be proud of what you do. Do not worry whether your solution is like anyone else's.

The projects at the end of this book may seem to be easy. . . but do not be deceived into thinking that you can skip them. After all, if they are easy for you, than it will not take long to do them.

Good luck!

Henry A. Taitt

Henry A. Taitt
Director

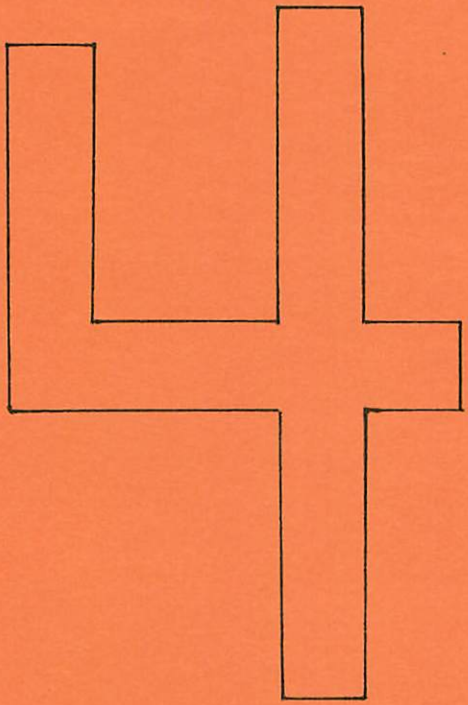
ORANGE PROJECT 1

Can you CREATE a computer program that will LIST in two columns the even numbers from 2 to 98?

When you have a program that works, copy a LISTing below and submit it to your teacher.

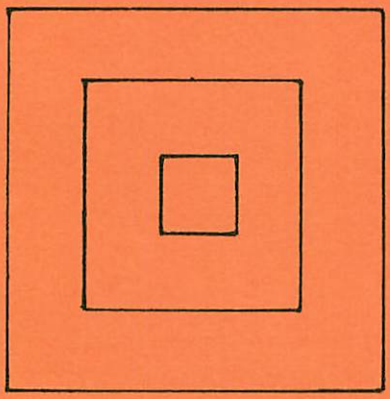
ORANGE PROJECT 2

Can you CREATE a program that will display on the screen a LARGE 4???



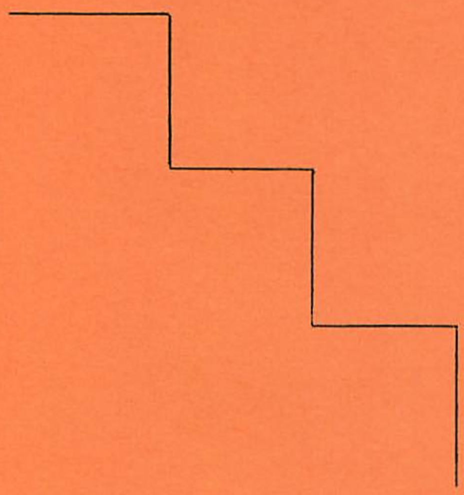
ORANGE PROJECT 3

Can you write a program that will draw a square bull's eye? When it is working show it to your teacher.



ORANGE PROJECT 4

Can you CREATE a program that will draw a set of stairs?



ORANGE PROJECT 5

Can you write a program that will cause your name to be printed with a box around it? Make the box as decorative as you wish.

You many wish to save this program so that you can use it at the start of any programs that you write to let people know you CREATED the program.

ORANGE PROJECT 6

CREATE a program that will draw a castle on the screen. Make it a good castle with your name or initials somewhere on the castle. Save the program on tape. Ask your teacher or a more advanced student to make a LISTing of the commands on a line printer. If they can, have them also make a line printer copy of your screen. (They can use your tape to load the computer that is connected to the line printer.) Send us the LISTing (and a copy of the castle if possible) and we'll send you your PROGRAMMER II card.

The cost of creating for you a PROGRAMMER'S card is included in the cost of this manual. By sending in this colored page, we know this cost has already been paid. If you are sharing this manual and cannot remove this page, then you may also receive your PROGRAMMER'S card and have your name listed in the newsletter by sending us a copy of your working program for this project along with $2.00 and a self-addressed, stamped envelope.

Send to:

Henry A. Taitt
CREATIVE Programming, Inc.
604 Sixth Street
Charleston, IL 61920

TI-99/4A

Your name ____________________

Phone # ____________________

Address ____________________

City, State ____________________

Zip ________ Birthdate __________

Don't forget to enclose a self-addressed stamped envelope.

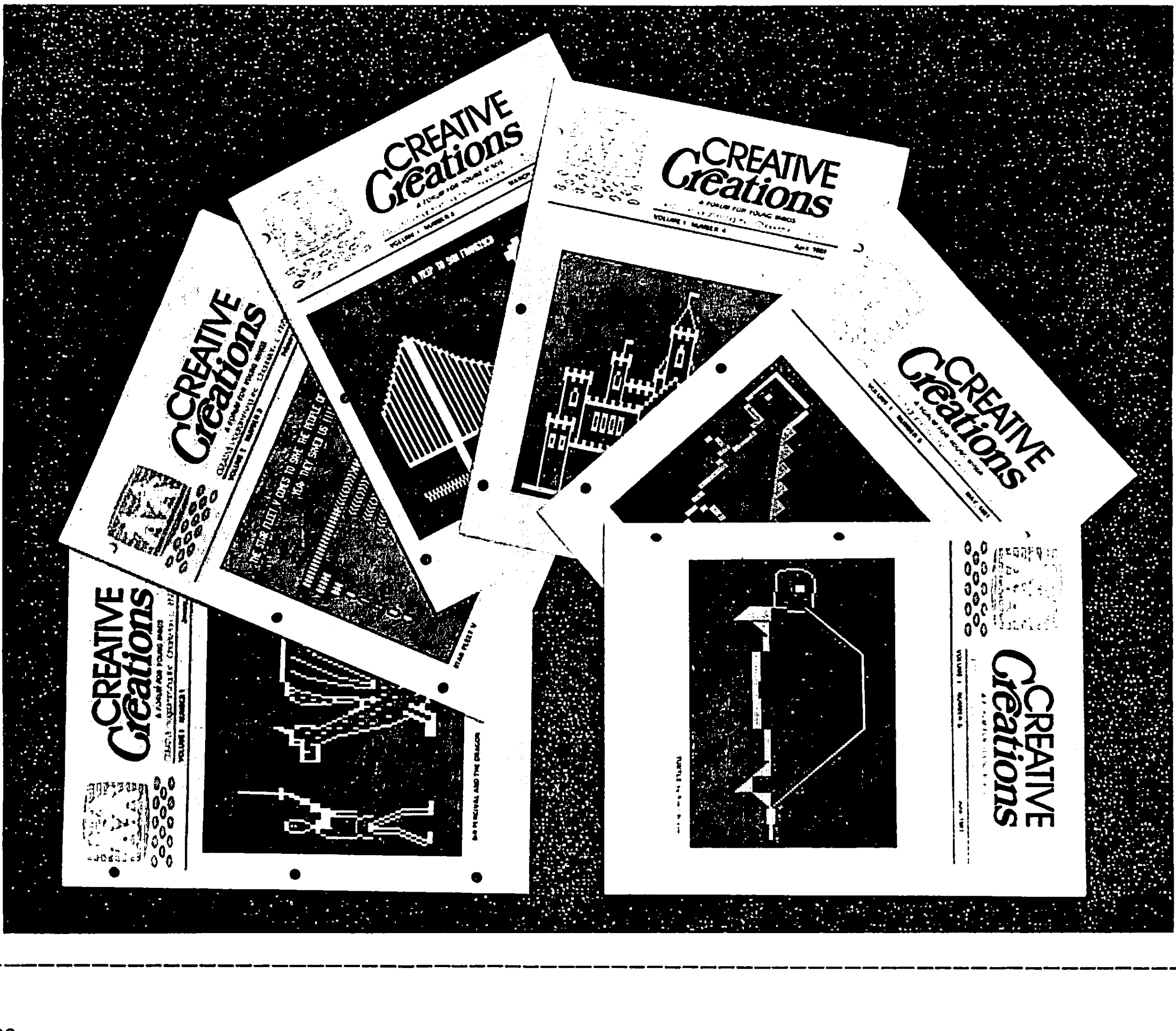
CREATIVE Creations

A FORUM FOR YOUNG MINDS

CREATIVE Programming, Inc., Charleston, IL 61920

A newsletter published 12 times a year. The articles are for young programmers, about young programmers and often written by young programmers.

Each month a graphics program created by a student is selected for the cover. It could be yours! Contests, mind bending challenges, computer game reviews, new creations, programs, even an X-rated column for parents and teachers who are running programs in their areas.



Name____

Address ____

City __________________________State _______ Zip_______

Please make checks payable to: CREATIVE Creations
604 Sixth Street
Charleston, IL 61920

Only $18 a year ($32 for two years) brings all twelve issues to your door. Join us today in sharing in the excitement of CREATIVE Programming through CREATIVE Creations.

☐ one year ($18.00) ☐ two years ($32.00)

CREATIVE PROGRAMMING INCORPORATED
A SUBSIDIARY OF R.V. WEATHERFORD CO.
604 6th St., Charleston, IL 61920

TI9-R-02